**NP completeness**

P: solved in polynomial time

NP: solved by non-deterministic turing machine in polynomial time

(P is a subset of NP)

**A decision problem L is NP-complete if:**

- L is in NP

- Every problem in NP is reducible to L in poly time

- Not able to find an algorithm, only a lookup table:

you can only come up exponential time approach which is impractical. If you don’t know about NP-Completeness, you can only say that I could not come with an efficient algorithm. If you know about NP-Completeness and prove that the problem as NP-complete, you can proudly say that the polynomial time solution is unlikely to exist.

A problem X is NP-Complete if

1. X is in NP, and

2. For any problem Y in NP, there is a "reduction" from Y to X: a polynomial-time algorithm that transforms any instance of Y into an instance of X such that the answer to the Y-instance is "yes" if and only if the answer X-instance is "yes".

If you can find a poly algorithm for X, then the entire class (which can be reduced to X) is poly

**A is reducible to B:**

A is poly reducible to B if A can be solved by calling B, and the work code from B is in poly. If B is solved in polynomial time, A is solvable in poly.

input A --> poly --> L --> poly --> output A

Optimization path

Decision problem:

given undirected weighted graph, does it exist a tree of weight <= W ==> L

**Hamiltonian cycle**

Hamiltonian Path or HAMPATH in a directed graph G is a directed path that goes through each node exactly once. We Consider the problem of testing whether a directed graph contain a Hamiltonian path connecting two specified nodes, i.e.

HAMPATH = {(G, s, t) | G is directed graph with a Hamiltonian path from s to t}

NP (Nondeterministic polynomial) languages are L that there is X belongs to L, then in ND poly.

-->can convince X belongs to L

NP hard

L --> poly --> L --> poly --> L'

NP-complete is reducible to NP-hard

P is in NP

**SAT problem**

sat --> G(poly) --> Hamiltonian --> NP complete

Check all possible assignments: 2^n, exponential time

Prove SAT is NP complete:

1. SAT is in NP

2. Prove for clique

3. Prove Dominating set

4. Prove vertex cover.

V.C.

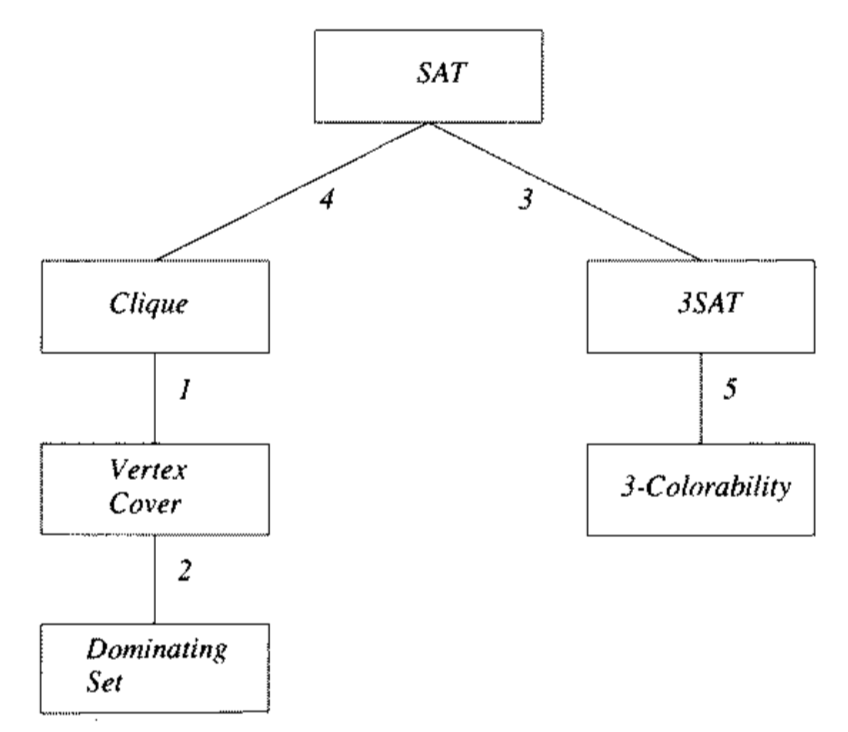
Dominating

set

G,k

Yes/No

G’,k’



**Clique is NP complete:**

Given undirected graph G and number k, does there exist a clique of size >= k. Between every pair there is an edge.

We want to prove: |clique| == m <-- SAT, |clique| == m --> SAT

**Vertex cover is NP complete:**

A vertex cover of G is a set of vertices such that every edge in G is incident to at least one of these vertices.

G,k --> G',k' --> vertex cover --> Exist a vertex cover G' <= k'? Yes/no

Prove: vertex cover implies there exists clique

In G', n-k is a vertex cover. There is no edge with both two vertices outside the vertex cover. In G, for every two vertices without an edge in G', there is an edge in G. Therefore, there is edge for the vertices outside n-k vertex cover. Then there is clique of size>=k in G.

no vertex cover implies there is no clique

G''=G

G', k' = n-k

min v.c. >= n-k <- clique < k

**Dominating set is NP complete**

every vertex of G is ether in D or is adjacent to at least one vertex from D. Is |s| <= k?

A vertex dominates itself and its neighbors.

**3SAT problem**

* Boolean expression in which each clause contains exactly three variables.

SAT

Clique

Formula

Yes/No

G,k

3SAT

3 color

G,k

Yes/No

G

Traveling Salesperson (TSP)

A cycle that visits every vertex of a graph once.

cost of tsp:

random assignment

Dynamic programming

Greedy

Divide and conquer

Graph

NP completeness: reduction from A to B

Hamiltonian cycle is NP complete.

TSP approximation